texSketch: Active Diagramming through Pen-and-Ink Annotations

Hariharan Subramonyam[1], **Colleen Seifert**[2], **Priti Shah**[2], **Eytan Adar**[1]
[1]School of Information,   [2]Department of Psychology
University of Michigan
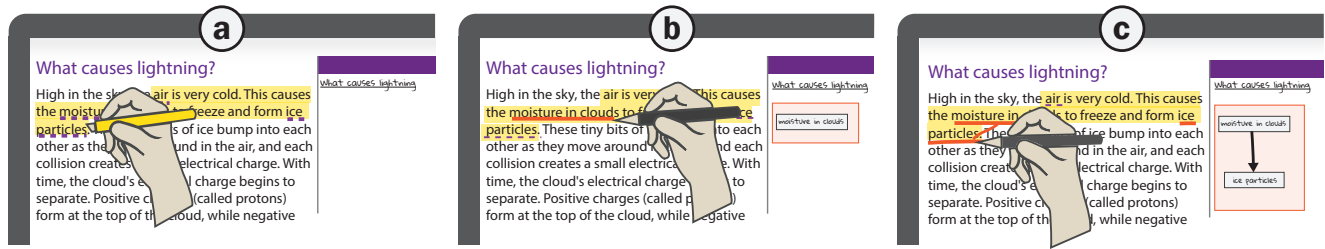Ann Arbor, MI
{harihars,seifert,priti,eadar}@umich.edu

Figure 1. Active Diagramming using texSketch: (a) The reader highlights an important phrase in the text, and system displays hints about key causal terms in that text. (b) The reader then adds those terms to the diagram view using underline ink annotation. (c) finally the reader connects the cause and effect terms by drawing a link across them in the text view. The connections are reflected in the diagram view as causal links.

ABSTRACT

Learning from text is a *constructive* activity in which sentence-level information is combined by the reader to build coherent mental models. With increasingly complex texts, forming a mental model becomes challenging due to a lack of background knowledge, and limits in working memory and attention. To address this, we are taught *knowledge externalization* strategies such as active reading and diagramming. Unfortunately, paper-and-pencil approaches may not always be appropriate, and software solutions create friction through difficult input modalities, limited workflow support, and barriers between reading and diagramming. For all but the simplest text, building coherent diagrams can be tedious and difficult. We propose *Active Diagramming*, an approach extending familiar active reading strategies to the task of diagram construction. Our prototype, texSketch, combines pen-and-ink interactions with natural language processing to reduce the cost of producing diagrams while maintaining the cognitive effort necessary for comprehension. Our user study finds that readers can effectively create diagrams without disrupting reading.

Author Keywords
Active Reading; Diagramming; Pen-and-ink gestures;

CCS Concepts
•**Human-centered computing** → **Pointing devices; Natural language interfaces;** *Scientific visualization;*

INTRODUCTION

When learning from text, readers develop meaning by merging sentence-level information into a coherent mental model [31]. For example, imagine reading a simplified explanation of 'lightning' (see Figure 1). What connections would you need to make in order to recall, paraphrase, or apply what you read? First, you would read about what happens at high altitudes, where moisture in the cloud freezes to form ice particles (connection 1). You would then find out that the particles collide against each other to create a charge (connection 2). Building on your understanding, you would encounter the last step: the negative charges on the cloud flow towards the positively charged surfaces on the earth (connection 3)–thereby creating a zap of lightning. Arriving at such a model requires you to (1) identify key concepts from raw text, (2) establish connections between those concepts, (3) make gap-filling inferences (e.g., protons on Earth cause charge separation in clouds), and (4) integrate prior knowledge with the new information that you are reading [32, 16].

As the complexity of text increases, building these connections accurately and persistently in the mind can be challenging. When reading STEM-focused text, learners often encounter complicated text-structures and difficult vocabulary. The constructive process may challenge their working memory, attention, and other resources [19, 37, 59]. Consequently, the reader may employ a 'low effort' approach, one that only utilizes readily available information. For example, a student may fail to understand that electric charge consists of electrons and protons which separate in the clouds. The result is a fragmented understanding of the text where the reader misses essential connections (or builds the wrong ones) [19].

Knowledge externalization strategies such as active reading, note-taking, and graphical modeling alleviate some of these difficulties. These strategies allow readers to record their

thought process using *persistent* and *manipulable* representations [34, 20, 16]. For example, readers employing *active reading* use annotations to 'emphasize' key concepts in text [1]. When *taking notes* as part of digesting the text, readers can reorganize the annotated concepts using structures that are meaningful to them (e.g., lists, outlines, etc.) [46]. Creating models by diagramming allows the reader to further integrate concepts into a cohesive mental model [21]. For this reason, graphical representations are often more effective when compared to simple note-taking [54]. Figure 1c illustrates a simple causal diagram, where the reader has connected a cause to an effect by a link (node-link diagram) [3]. Notably, even if the *author* provides a diagram, there is learning value in 'forcing' the reader to produce the externalization [77].

Existing externalization strategies scaffold different steps in the comprehension process: *selection*, *organization*, and *integration* [46, 16]. However, few tools support *all* from within the same reading workflow. Use of paper and pen for highlighting and simple text marking (selection) is fast and effective, but organizing and integrating diagrams is not. Making a diagram requires iteration, for which paper is ill-suited (e.g., adding text, moving blocks, adjusting structural layouts). Digital reading tools feature highlighting and annotation support for word– and sentence-level tasks. Unfortunately, annotations and markup of this type do not help the reader in building higher level connections. With few exceptions (e.g., [71]), note-taking tools make the text presentation primary; drawing, or note-taking surfaces are secondary and disconnected (e.g., [56]). In contrast, diagramming tools such as CogSketch [25], emphasize the drawing canvas, limiting the usefulness of annotations on the text display for diagram construction. Reading complex text requires multiple reading cycles and seamless switching between different types of representations–preferably without high switching costs. In fact, the effort required to map verbal representations into visual elements can outweigh its benefits [73]. Thus, the motivating question for our work is: How can we facilitate externalization across different levels of understanding (concepts, connections, and coherent mental models [33]) without burdening the reader?

Through a design probe on diagramming, we found that (1) readers require a flexible mechanism for switching between reading and diagramming tasks, (2) drawing is time-consuming and includes redundant tasks such as copying text concepts to the diagram view, and (3) diagramming is iterative and hierarchical, where low-level representations (e.g., small fragments of the causal diagram) can be merged upwards. On this basis, we propose *Active Diagramming (AD)*, a generative learning strategy in which readers construct diagrams through active reading and information organization tasks. As the end-user reads and annotates text in our system, *texSketch*, the annotations are transformed into diagram elements. Diagram elements can be created, modified, and combined by the reader through simple pen-and-ink gestures on both the text and diagram windows. Automated NLP-based tagging offer 'hints' to the reader about possible externalization targets in the text. This tagging can aid—but does not limit—the reader.

TexSketch demonstrates a way of supporting essential comprehension tasks. We work to balance the efficiencies of new interaction techniques and automation with the cognitive benefits of creating the diagram [21, 77]. Both the process of creation as well as the completed artifact (the diagram) support comprehension, review, and communicative functions. Our key contributions include: (1) understanding *active diagramming* as an approach for constructive reading, (2) an interactive tool to support AD, and (3) an extensible architecture for applications that connect text and visualizations.

RELATED WORK
According to the construction-integration model of text comprehension [33], reading happens through *multiple levels* of mental representations. Readers first recruit key concepts from text into working memory. They then formulate sentence-level 'propositions' relating those concepts. Finally, the reader integrates those propositions into a coherent model. To do so, they execute a number of cognitive operations, including: (1) search for relevant concepts, (2) delete or combine redundant concepts, (3) substitute and generalize specific terms with superordinate terms, and (4) correct erroneous connections [12]. As they accumulate more information, these operations become harder for the reader. Readers spread their attentional resources more thinly. In response, they can recruit additional resources through increased concentration or through external aids [20, 34, 36, 63]. When effective, the result is a mental model that captures the complexity of intended meaning. Prior research has looked at ways to offer external scaffolding. Broadly, these techniques include: (1) selection, (2) organization, and (3) integration aids [46]. Tablet devices with stylus are particularly effective for active reading [51] and a number of systems (including our own) focus on this platform.

Selection Aids
Writers create attentional aids for readers through typographical signals such as titles, section indicators, bold and italicized text. However, writers cannot anticipate all needs of different individuals, and excessive signaling cues may overwhelm readers [41]. Annotation tools allow readers to create their own typographical cues to support comprehension [49]. In active reading, placemarks, anchors, structural, and attentional cues [45] reduce working memory load, allowing the reader to better engage in analysis and synthesis. By studying active reading behavior on pen-and-paper [30], a number of studies have looked at enhancing the annotation experience by making them searchable [56, 27, 29], automatically categorized [67], or more organized [5]. These enhancements make it easier for readers to 'reactivate' previously read information by quickly searching or scanning through their annotations. In texSketch, we use similar ink annotation techniques to help readers select key concepts, and then use those annotations as building blocks for higher level representations.

Organization Aids
Organizers such as lists, outlines, and blocks are helpful when aggregating and summarizing related concepts. To support this, one approach is to have a system automatically process selections to generate an initial organization scheme [56, 64, 11].

With texSketch, we automatically add selected text into sub-diagrams. There are different places where we can 'clean up' the diagrams–either in tight integration with selection (e.g., [6, 71]) or in a 'post-processing' step [28]. We opt for the latter. texSketch adds all selections to diagrams and allows readers to remove redundant annotations at a later stage. This is similar to GatherReader [28], which supports a flexible workflow via *deferred* action. To determine which sub-diagram to add the selection to, we use color-coded annotations inspired by "Linking-by-inking [55]." The system creates hyperlinks between instances of the same annotation symbols, thereby combining selection and linking from within the same action. To link selected text, we use simple ink gestures [9, 50].

Integration Aids

Integration requires combining individual concepts and relations to synthesize meaning. According to the cognitive theory of drawing construction, readers may translate verbal text to visual information by selecting, organizing and then integrating [72]. Graphical representations such as concept maps, flow diagrams, and tree diagrams can be used to explicitly depict the structural and conceptual relationships found in text [60, 3]. By generating diagrams themselves, readers are able to leverage spatial memory to develop a coherent understanding of the text [26, 16, 17]. However, diagramming tools are largely stand-alone and disconnected from source information [23], making it hard for readers to successfully create diagrams while reading [73]. In texSketch we address this gap by directly integrating selection and organization annotations with diagram construction.

We draw inspiration from sketch-based diagramming tools to inform the design of ink interactions for texSketch. One approach uses pre-defined layout conventions. This allows the reader to add, delete and connect different diagram elements through ink interactions in constrained layouts [42, 10, 14]. Other approaches focus on automatic transformation from set expressions, equations, and logic descriptions into Venn diagrams, geometric representations, and vector representations [18, 65, 8]. In texSketch we combine both approaches to automatically transform selection annotations into diagram elements, and use direct manipulation to visually create relationships between those elements. We also employ natural language processing to identify concepts and relations [47, 35]. However, instead of automatically creating diagrams, we offer visual hints to support diagram construction.

Previous studies have looked at layout configurations for active reading tools. Design options include side-by-side views [71], white space expansion [78], and multiple connected displays [15]. In texSketch, we opted for a side-by-side layout for text and diagrams as it allows easy coordination and navigation between different pieces of text and the diagram. Inspired by past work, we use stylus-based ink gestures to trigger different actions and thereby maintain consistent interactions across the text and diagram view [7, 39, 70]. The diagrams in texSketch are rendered using a sketchy (hand-drawn) style. In contrast to high-fidelity graphics which come off as 'finished,' this approach has been shown to promote user engagement and direct manipulation [75].

A DESIGN PROBE FOR ACTIVE DIAGRAMMING

We conducted a probe study to explore features that might aid AD and to better model the scaffolding behaviors and needs of readers. To probe reading behavior when coupled with diagramming, we designed the study as an A-B-A design [13]. That is, readers first made diagrams using pen and paper (A), followed by use of a digital prototype with baseline features implemented (B), and finally returned to the paper condition (A). Feedback was collected at each stage for analysis. This allowed us to understand how end-users might produce diagrams on pen and paper, elicit new feature requirements for the digital tool, and finally allow participants to reflect on the differences between tool and paper to provide comparative feedback and additional feature suggestions. The *baseline* prototype (as distinct from the full texSketch) implemented just the ink-and-link feature (supports adding nodes to the diagram by underlining, and linking with a pen stroke). It was designed to function similarly to basic digital note-taking software with support for text selection and extraction, and space next to the main text for notes and diagrams (e.g., [22, 44]).

Procedure

We recruited five participants (three graduate students, and two industry practitioners). All had prior experience using visual note taking with pen and paper (e.g., one participant reflected on the challenges in making symbolic logic diagrams in a philosophy course using a look-up table for symbol mapping). Each individual session lasted a total of 60 minutes with four AD tasks, and $15 compensation. Each participant worked with two texts on *paper* and two with the baseline prototype. In order, the topics were: *What causes rain?* (paper), *What causes lightning?* (baseline), *How do engines make cars move?* (baseline), *What causes coral bleaching?* (paper). The topics were at a high school level (by Flesch-Kincaid score), with an estimated one minute reading time.

At the start of the session, the study coordinator provided a brief overview of AD and drawing causal diagrams. Once participants indicated familiarity, they proceeded with the first pen and paper task. Participants received a letter-size handout (landscape layout) with the text explanation on one half of the paper, and a blank surface for drawing on the other. We instructed participants to engage in AD as if they were creating a diagrammed explanation of the text in preparation for a test. Participants had access to colored pens and highlighters for drawing, and were asked to indicate where in the text the corresponding nodes came from. The digital prototype was deployed using Microsoft Surface Book in tablet mode, 13.5-inch, 3,000 x 2,000 Display. For the tool stage (B), participants were given a guided tutorial of the ink-and-link feature. At the end of the study, participants provided feedback through a semi-structured interview [52]. The sessions were video recorded for analysis.

Findings and Design Considerations

We analyzed 20 diagrams (5 participants × 4 diagrams), interview notes, and video recordings. Videos were coded for different AD activities including reading, adding a node, adding a link, editing, and making a new diagram. At a high level,
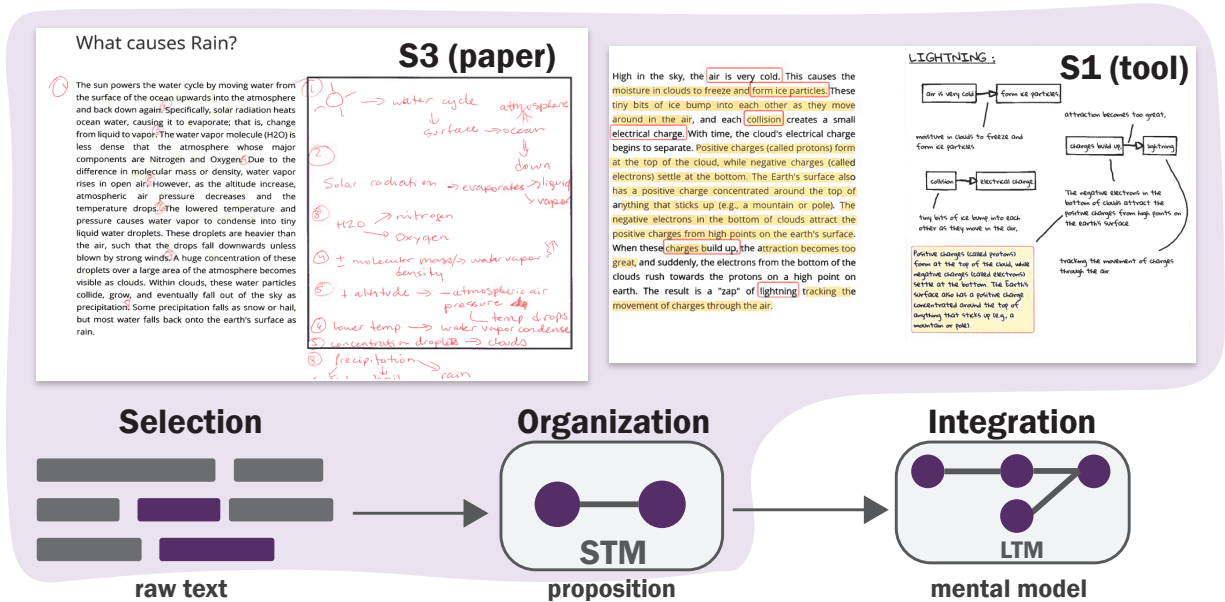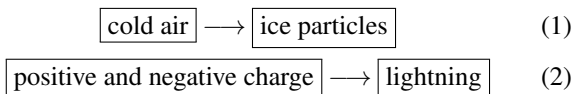
Figure 2. An overview of the text comprehension model. Raw text is recruited into short-term memory (STM) to form individual propositions, and later combined with earlier propositions from long-term memory (LTM).

we found three critical concerns for supporting better AD: *diagram representations, speed, and comprehension process.*

Diagram Representation: While participants understood causal diagram notation, most diagrams (paper and prototype) were limited to simpler proposition-level representations. For example, they might draw two separate causal relations and never integrate them:

$$\boxed{\text{cold air}} \longrightarrow \boxed{\text{ice particles}} \tag{1}$$

$$\boxed{\text{positive and negative charge}} \longrightarrow \boxed{\text{lightning}} \tag{2}$$

In most cases, the diagrams were organized one below the other in the order they occurred in the text (similar to a visual list). This aligns with Mayer's selection-organization-integration model for active reading [46], except we only saw three instances in which participants made connections across propositions. In these cases, manipulation was effortful, requiring participants to either make long connecting lines across immutable diagrams or redrawing based on updated understanding. To support the production of more integrated diagrams, we propose that AD systems allow the reader to easily connect new elements to their diagram or refine their diagram as they go. More concretely, tools should encourage readers to **establish connections between what they are currently focused on reading and what is already visualized (Design Guideline 1).** Additionally, AD tools should **provide integration affordances for easily manipulating and combining diagrams (D2).**

Speed: In the paper condition, we observed that participants took between 5-10 minutes for each text–much longer than the expected one minute. A driver of this was the ten second average time to create a node in the diagram. Furthermore, nodes

were imprecise and long (average of 4 words). With the prototype, node creation time was reduced to 1-2 seconds. However, *arranging* nodes and links was still hard. Participants had to determine where to position nodes across multiple smaller diagrams. On average, participants took 5 minutes to complete a diagram using the prototype. In order to maintain attention on reading, AD tools should assist readers in **quickly and precisely creating and arranging diagram elements (D3).**

Comprehension process: We observed two different approaches for interleaving diagramming and reading. The most common was to create diagrams after reading each sentence (Figure 2). However, this can be disruptive to reading. One participant noted: "[my] goal was to make the diagram rather than to really understand the material." In a second approach, participants read through the entire text while highlighting key sentences *before* creating the diagram. One participant mentioned regret at *not* using this approach: "I am unsure if doing it at the low level I did was effective. . . should have had higher level nodes. . . I created sentence level nodes." Based on these findings, we propose a three-step process for AD: *read (select), then draw (organize), then connect (integrate)* [46]. While it may be necessary to switch between tasks, too much switching may be inefficient. Cognitive, informational, and working memory limits may be different between readers and so the optimal timing of the context switch may be different as well. We also note that the 'Reading ahead' strategy allows readers to anticipate which nodes they might draw and where. Depending on the length and structure of the article, the iteration may happen after reading a paragraph, page, or even article levels but preferably, not each sentence. Tools for AD should offer **flexibility in switching between these three different steps, but also guide the reader in taking these steps in order and sustaining attention on each (D4).**
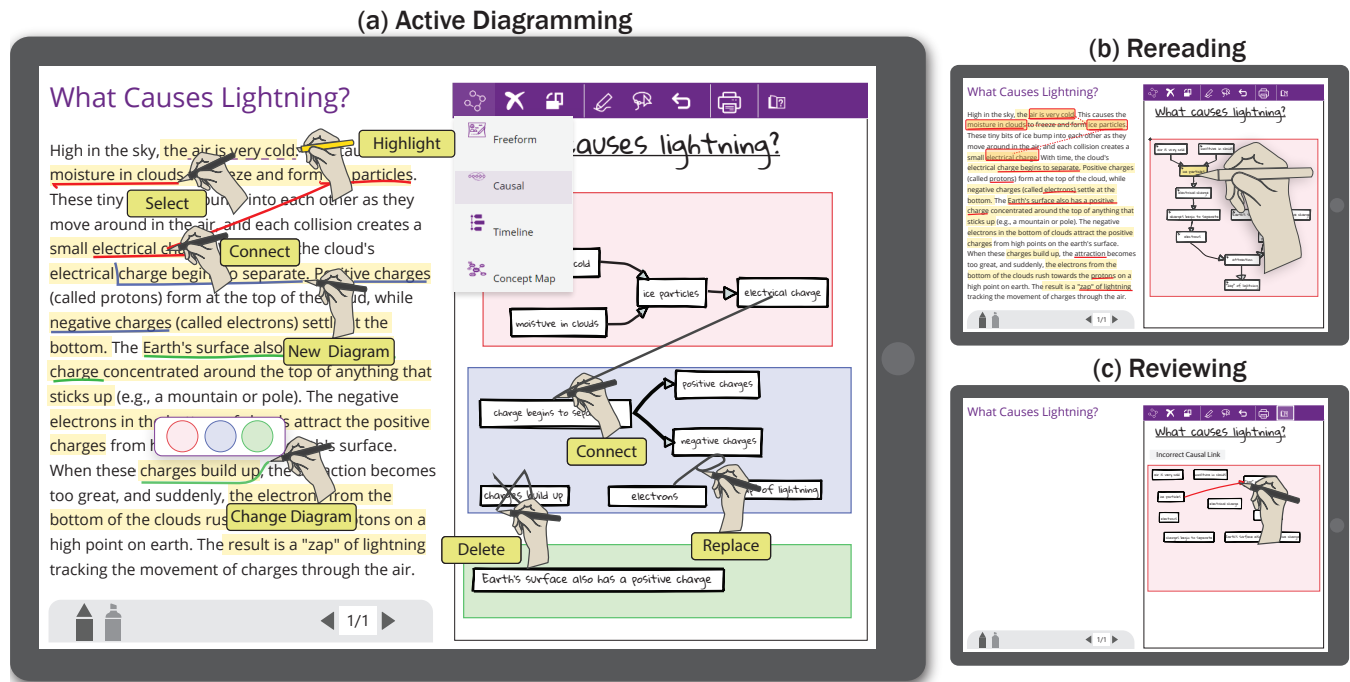
(a) Active Diagramming

(b) Rereading

(c) Reviewing

Figure 3. Active Diagramming using texSketch: (a) selection and organization using ink gestures, (b) re-reading using integrated diagram, and (c) reviewing by reconnecting nodes from memory.

ACTIVE DIAGRAMMING WITH TEXSKETCH

Driven by probe, we implemented texSketch (Figure 3). texSketch's interface is divided into two main regions: a text view on the left, and a diagram view on the right. The *text view* supports direct ink annotations using the pen and highlighter tools which are accessed from the bottom toolbar. The toolbar also contains page-navigation options for multi-page text. The *diagram view* is a zoomable, scrollable canvas with support for touch interactions over diagram elements. Readers can directly draw on the canvas, and interact with diagram elements through direct manipulation gestures (touch and pen-based near-touch interactions). The diagram toolbar has a menu at the top for selecting the diagram type, deleting the diagram, selection tools, undo, print, and review. To better understand how texSketch supports AD, we follow Tanya, a high school student reading about the mechanism behind lightning.

Set-up

Tanya opens texSketch on her tablet device. Although she can load a PDF document directly, in this case the lesson about lightning is already in the list of available topics. Selecting the lesson takes her to the main interface with the lightning text loaded in the text view. Because Tanya's goal is to understand the causal phenomenon, she taps on the 'causal' diagram option in the *diagram type* drop-down menu. texSketch adds an empty causal diagram to the canvas (a rectangle area) which is assigned a unique color. Because texSketch allows readers to add multiple diagrams to the canvas, we use color-gestalt to match the color of the pen tool with the color of the current active diagram (aligned with design guideline D1).
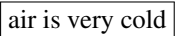
Pre-reading and Highlighting

Tanya selects the highlighter tool from the bottom toolbar and begins reading the text. As she reads, she highlights important sentences that contain key concepts about lightning formation. Using a text understanding model, texSketch provides *hints* about key causal terms in the highlighted text by adding a dashed underline on those terms (D3). Tanya is free to ignore these on-demand hints or explicitly ask for them by highlighting a text region. This design encourages the two-pass reading approach suggested by participants as potentially better. The NLP features are also motivated by our probe study. We found that readers are not always succinct in their underlining behavior. This may be for a good reason–the reader may remember a particular concept (e.g., a noun such as 'sun') in the context of adjectives and verbs (e.g., the hot sun glows). However, this may also be a characteristic of underlining behavior in which the reader marks the text as a way of keeping place in their reading. A consequence is that nodes that *should* be combined in the diagram can't be. For example, if the text also has, 'the clouds cover the sun ' it may be difficult to integrate the two sun nodes ('hot sun glows' and 'sun') as they are semantically different. While we do not want to prevent the reader from creating structures that are meaningful to them, the (subtle) hints that texSketch provides may lead to precise diagrams.

Active Diagramming

After highlighting a portion of the text about ice particle formation, Tanya decides to model the causal relationships in that portion of the text using AD. texSketch offers Tanya the flexibility to interleave reading and diagramming activities based

on her preferences and abilities–whether this is at the level of a single sentence, a paragraph, or even the entire document (D4). In AD, Tanya engages in analytical (thorough) reading while also constructing causal diagrams. Her workflow involves *selection, organization,* and *integration* features.
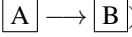
Selection: Tanya rereads the previously highlighted text, paying close attention to potential causal phrases in each sentence. Using the pen tool, Tanya *selects* individual causes and effects to add to the diagram by making an underline annotation. As she underlines the phrase "air is very cold," texSketch automatically adds it to the diagram as a causal node (i.e., text with a bounding rectangle box: air is very cold). Similarly, Tanya adds the phrase "moisture in clouds" as a cause which results in the effect "ice particles." If the phrase spans across multiple lines, Tanya can underline across multiple lines of text, and texSketch automatically combines those words into a single node by identifying punctuation marks such as hyphen or period. When this is not desired, Tanya can split the phrase into two nodes by drawing a vertical split annotation ('|') in the text or on node (D2). Throughout this selection process, the NLP-based hints provided by texSketch assist Tanya in choosing the right causal concepts to diagram (D3).

Organization: After adding nodes to the diagram view, Tanya can organize them as a diagram by creating directional edges from causes to effects. To do this, she first draws a line annotation between the words "air is very cold" and "ice particles" in the text view. But, she can also do this from the diagram view (D4). Based on the direction of the line, texSketch interprets the node at the starting position as the cause and the end point as the effect. In the diagram view, the nodes are automatically connected by a directional arrow (i.e., air is very cold $\longrightarrow$ ice particles). To reduce clutter, texSketch converts the hand-drawn links on the *text view* into a dotted line. A system setting also allows Tanya to decide if the line should persist or fade out. However, faded links become visible when she is hovering over the connecting text with her stylus (D1). Further, to reduce burden on the reader, texSketch automatically updates the diagram layout (D3). Diagrams are oriented left-to-right, but readers can change to a vertical layout using the menu toolbar.

In addition to organizing nodes into causal diagrams, texSketch allows Tanya to create separate (sub) diagrams for each reading cycle. When reading the text about earth's positive charge, Tanya wishes to separate out these causal relationships from the ones about clouds. To accommodate this, texSketch supports a variant of the selection (underline) ink gesture. To select a node into a new sub-diagram (instead of the current active diagram), Tanya draws an L-shaped underline annotation around the text (e.g., "mountain or pole"). This creates a new sub diagram, which is assigned a unique background color, and the node is added to that diagram. The color of the pen tool also changes to the diagram color to indicate that subsequent selections will be organized within the new diagram. Then, to accommodate for complex text structures, Tanya can switch between existing diagrams. To do this, she selects the text by drawing a second variant of the underline gesture, one with a diagonal stroke ending. A context menu

appears that lets her change the target diagram for that text selection using a set of color-coded labels corresponding to the diagrams currently on the canvas. In this manner, Tanya continues to read the entire text across multiple reading cycles. In each reading cycle, she creates and organizes the causal relationships into one or more sub-diagrams.

Integration: At this stage, Tanya integrates the various diagram pieces into a causal model for lightning. That is, she constructs an understanding by connecting propositional representations into an integrated mental model of the text. In the *diagram view*, she combines the sub-diagrams for earth and cloud by drawing a connecting line between the nodes for 'Protons' and 'Electrons.' Drawing a line between two nodes across different causal diagrams will directly append the second diagram (node at line ending) to the first diagram (D2). To combine nodes, Tanya can draw a line with a loop at the end (e.g., combining the two nodes 'negative charges' and 'electrons') to signal co-reference. NLP techniques are not always accurate, so the loop gesture allows the reader to control the integration process. In science text it is common to interchangeably use general terms (e.g., 'algae') with specific instances (e.g., 'zooxanthella'). The loop annotation allows readers to merge such generalizations at the diagram level.

For complex explanations, Tanya can also create nested causal relationships. This is useful when she has abstracted a causal mechanism as a simple edge. For example, *A* causes *B* (i.e., A $\longrightarrow$ B), but the link '$\longrightarrow$' is actually an abstraction of a more complex mechanism represented in a sub-diagram *D*. She can integrate the secondary diagram by drawing a new line from it to the link. This combines the two diagrams by nesting the diagram *D* at the position of the link. Throughout the diagramming process, Tanya can delete any node or link by marking an '$\times$' annotation over it with the pen (similar to scribble erasure). In this manner texSketch supports the final step of the **selection-organization-integration** workflow. Tanya's final diagram is shown in Figure 3b.

Rereading and Reviewing
In addition to potential learning benefits from AD, these diagrams serve as rich artifacts for rereading and reviewing. With its tight integration between text and diagram elements, texSketch offers interactive assistance for these tasks. When Tanya revisits her diagram to prepare for an exam, the diagram serves as a visual table-of-contents. She can hover over a diagram node (using near-touch pen-tracking interaction) to retrieve additional details from the text. texSketch highlights the corresponding text in the text view. Because the causal diagram forms a 'skeletal' network structure underlying the text, Tanya can use her stylus to trace through causal structures. As she hovers over a node in the diagram, texSketch highlights both the source text and conceptually neighboring text (based on the network neighbors of the node in the node-link diagram). For *multi-page* text, the diagram elements corresponding to that page are highlighted as she moves between pages. These interactions support rereading by allowing Tanya to switch between summary (the diagram) and detail views (the text).

To develop a deeper understanding of the text or to self-test her understanding, Tanya can enable the review mode from the
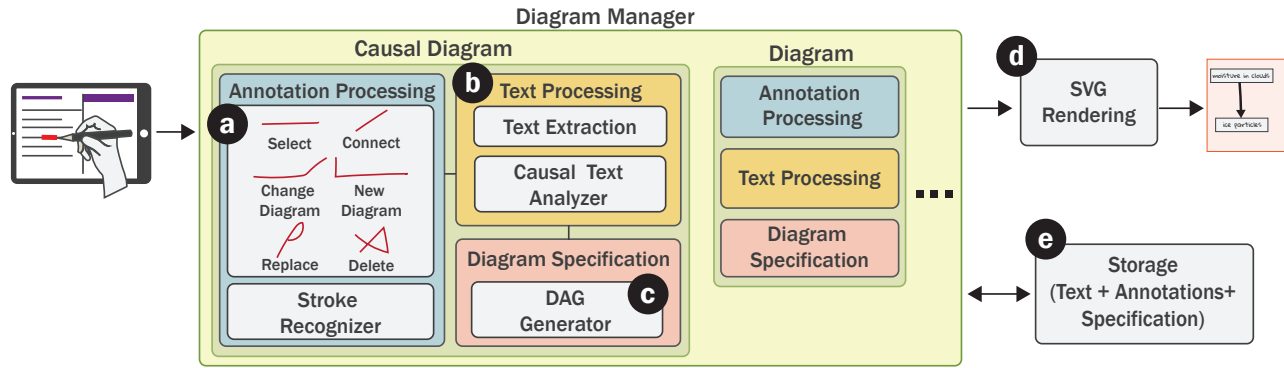
Figure 4. texSketch System Architecture: (a) Annotation processor identifies ink gestures, (b) text processor fetches text context and NLP labels. (c) Diagram generator creates the specification, and (d) a SVG renderer updates the diagram canvas. (e) Text and annotations are stored for later use.

diagram toolbar. For causal diagrams, the review mode will hide the text view and remove the links between nodes (see Figure 3 c). She can then attempt to redraw the edges from memory. texSketch provides visual feedback when she makes erroneous connections. texSketch was designed to support future extensions for review functions (e.g., blanking nodes, offering distractors, implementing spaced learning, etc.).

SYSTEM ARCHITECTURE

At a high level (see Figure 4), texSketch combines ink annotations with associated text content to create diagram elements (i.e., $Text + Annotation \longrightarrow Glyph$). When the end-user makes an annotation, the annotation processing module corresponding to the current diagram type (Fig. 4a) fetches the diagram 'operation' to be executed from a look-up table. It then calls the text processing module (4b) to retrieve the text inputs for that operation. Finally, the diagram generator module (4c) executes the diagram operation and outputs a SVG specification which is rendered on the canvas by (4d) SVG rendering module. We implemented texSketch as a web-based server-client application. The server is implemented using Node.js and is responsible for tasks such as text analysis and storage. The client, written using HTML and JavaScript, renders annotations and diagrams as SVG elements. While complete details of our implementation are beyond the scope of this paper, we describe the main components of texSketch.

Annotation Processing

The main function of this module is to identify the annotation made by the end-user and retrieve the corresponding diagram operation. It consists of a stroke recognizer and a look-up table with unique annotation labels that map to specific diagram operations. The design space for annotation mapping includes stroke patterns (e.g., [69]) as well pen specific attributes (pen type, stroke color, thickness, etc.). The diagram operation is defined as a function that takes in one or more predicates and outputs a diagram specification. For example, for causal diagrams, the underline annotation maps to the operation ADDNODE(*nodetext*). The stroke recognizer module is implemented using the NDollar recognizer [4]. It holds the training data as a collection of labeled annotations for each annotation stroke. Each data-item is represented as an array of points in the $x - y$ coordinate space and may optionally

include pen specific attributes. Lastly, this module handles 'cleanup' operations such as tidying hand drawn annotations, deleting stray annotations, and fading annotations.

Text Processing

This module retrieves the predicate values for the matched diagram operation by processing the annotated text. It consists of a text extraction sub-component and an optional NLP sub-component. For simple selections, the text extraction component finds the text that is closest to the annotation by calculating the $x - y$ distance, annotation span, and position of annotation corresponding to the text (above, below, on, etc). During extraction, it also considers annotation history to determine if the text is part of a multi-line selection. For more complex annotations such as linking two annotated texts, deleting annotated text, etc., texSketch identifies predicates by factoring in other parameters such as distance and overlap to other annotations and diagram elements.

Depending on the diagram operation and end-user needs, this module may also parse and tag the text by applying natural language processing. For example, the extracted text may be transformed by separating words based on punctuation. Specific to causal diagrams, we use a causal relation extractor to identify the causal terms within the highlighted text (implemented using SpaCy [2]). To extract causal relations, we first perform dependency parsing and coreference resolution using the NeuralCoref model [24]. Then, we apply a set of causal rules over the dependency tree (specified in [68]). We found that for scientific text, this was better at extracting cause and effect relations when compared to other approaches such as semantic role labeling. Phrases that may correspond to diagram elements can be subtly underlined in the text view.

Diagram Generator

Once the operation has been identified and the predicates determined (e.g., ADDNODE(*sun*)), this module implements the operation by creating, modifying, or deleting diagram elements. The output is an SVG specification that can be rendered onto the diagram view. This module also handles how different diagram elements are visually arranged. In our causal diagram implementation, we use mermaid.js [48], a markdown specification and SVG generator to create the nodes and edges. The generated SVG is rendered by the SVG
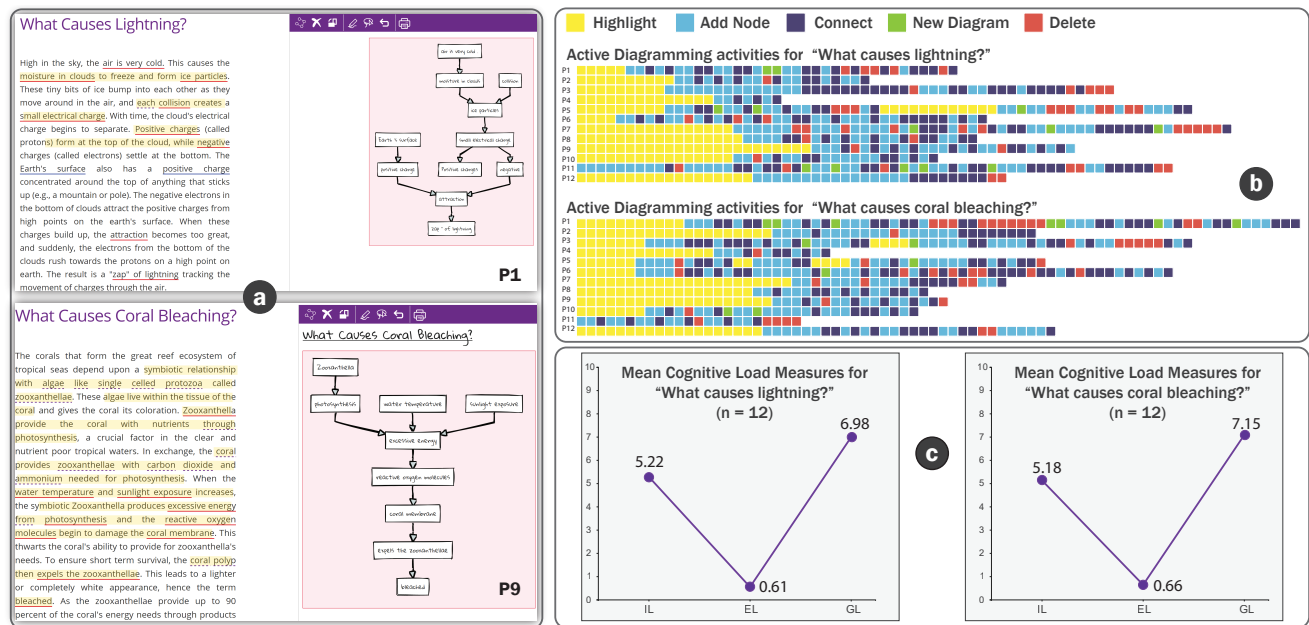
Figure 5. Evaluation Results: (a) Diagrams created by participants for Lighting (*top*) and Coral Bleaching(*bottom*). (b) texSketch log-data visualization showing different diagram actions, (c) Results from cognitive load survey (IL=intrinsic load, EL=extraneous load, GL=germane load).

rendering module. To provide for a hand-drawn appearance of diagrams, we use rough.js [66], to render sketchy lines. The generated diagram elements, along with the annotation and corresponding text are retained as JSON objects (which texSketch uses for rereading and reviewing operations).

EVALUATION

To evaluate texSketch, we aimed to determine whether readers are able to create causal diagrams from text using our AD approach. Specifically we wanted to gather feedback on (1) the experience of Active Diagramming while reading, (2) its effectiveness at producing coherent causal diagrams, (3) and overall usability of texSketch. We conducted an in-lab user study with the same tasks from our initial probe study. For this study, we recruited 12 participants who were graduate students or working professionals with prior experience in visual note taking. Each session lasted 75 minutes, and participants were compensated with $20 for their time.

At the start of each session, participants were given a brief overview of AD and how to create causal diagrams. The study coordinator offered a guided tutorial of texSketch using a new topic: *"Why do fireflies light up?"*. Once participants indicated familiarity, they proceeded to *practice* AD on their own using the topics for engines, and rain presented in random order. During this phase, participants could request assistance when needed. They then proceeded to work on the *test* tasks on lightning, and coral-bleaching presented in random order. After *each topic* in the test set, participants filled out a cognitive load survey (CLS) [38] to measure their subjective cognitive load during that task. At the end of the study, they filled out a usability questionnaire [43]. The study concluded with a semi-structured retrospective interview to gather feedback on their experience with texSketch. We captured all diagrams created along with texSketch log data for further analysis.

Findings

Active Diagramming during Reading: Based on our observations and participant feedback, encouraging the two-pass approach was effective. All but one participant (P11) began with highlighting actions, then added nodes, and then linked nodes (see Figure 5b). Further, most participants read and highlighted text in a single reading-cycle before proceeding to engage in closer reading and diagramming. Participants agreed that highlighting first provided them with the "context" for diagramming. P4 reported that they tended to highlight a lot while reading, and texSketch's two-pass approach helped them *"reduce things down."* In the second pass, participants found focused reading over highlighted text to create diagrams to be engaging and fun. In most instances, we saw that participants would underline concepts, and at a later time go back delete unnecessary concepts as they made causal links. According to P6: *"It was parked there and it really helped. I was able to track whether or not my thoughts made sense with what ever else that came up later."* We also observed that participants used the organization step in AD as a way of refining their initial propositions. For example, participants would first select "rise in water temperature and sunlight exposure" as a single node, and later break it down into two nodes when forming causal connections.

Participants appreciated that it was fast and effortless to edit and iterate over their diagrams while reading (P3: *"I really like the deleting. . . I think its just if I make a mistake its not a big deal I can just delete. . . "*). This validates texSketch's support for draft versus craft approaches [58] in which readers do not have to consider revising as effort wasted. From the log data, we can see that deletion tended to occur later in the AD process. P10 self reported a reading disorder with a tendency to jump back and forth across text. They found that texSketch and the AD process helped them work backwards to where they

need to be. They would start selecting from the bottom of the paragraph (the eventual effect) and then backtrack by asking "What caused this?" On average participants took 6.3 *minutes (SD = 2.9 minutes)* to complete the lightning task, and *6.8 minutes (SD = 2.7 minutes)* for corals. P1 was excluded from these summary statistics as an outlier, spending 20 minutes exploring the symbiotic relationship between corals and algae.

The CLS questionnaire uses a ten-point rating scale to measure (1) intrinsic load (IL) which corresponds to difficulty of the reading material, (2) extraneous load (EL) from the tool and reading intervention, and (3) germane load (GL) referring to students' self-perceived learning. Based on our analysis of the CLS responses, participants reported a high intrinsic load *(lightning: mean = 5.22, SD = 3.1, corals: mean = 5.13, SD = 2.2)* and a low extraneous load *(lightning: mean = 0.61, SD = 1.2, corals: mean = 0.66, SD = 1.5)* while using texSketch (see Figure 5c). Further, their rated self-perceived learning was high *(lightning: mean = 6.98, SD = 2.2, corals: mean = 7.15, SD = 1.9)*.

Diagram Coherence: To evaluate coherence of diagrams, we rated each diagram against a rubric. For each diagram, we awarded a single point each for every correct node and link. The diagrams matched, on average, 83% (Lightning) and 90% (Coral) of the correct nodes, and 78% (Lightning) and 93%(Coral) of the correct links. Participants were able to successfully integrate sub-diagrams in texSketch in contrast to the fragmented diagrams produced in the probe study. All but one participant (P5) created a single connected diagram for each text. Additionally, many participants created larger causal diagrams with an average of 9.8 (Lightning) and 7.5 (Coral) nodes. Because the study version of texSketch did not support direct labeling of links, participants devised a workaround by creating additional nodes as link explanations (e.g., $\boxed{\text{cloud moisture}} \longrightarrow \boxed{\text{freezes}} \longrightarrow \boxed{\text{ice}}$). This feature can be better supported in texSketch using alternative gestures and graphical annotations on nodes.

User Experience: All participants reported that texSketch was fun to use and they were able to quickly learn the ink gestures implemented. Multiple participants commented that they wished they had texSketch while in high school. By reflecting on their own note-taking experiences they clearly saw benefits to having an integrated reading, organizing and diagramming tool for learning (P11: *"Drawing relationships literally over the text is to build on the potential on digital text that others tools haven't done yet. . . ," and P2:"Visio makes me stressed out. . . you wouldn't use it casually"*. Based on the seven-point scale usability questionnaire, participants rated texSketch to be easy to use (*mean = 6.0, SD = 0.9*), and said they could easily recover from mistakes (*mean = 5.75, SD = 0.86*). They also rated it highly on learnability (*mean = 6.25, SD = 0.86*), but felt it could be improved to meet additional needs such as adding portions of images to the diagram or adding custom gestures (*mean = 4.3, SD = 1.15*).

Participants had mixed opinions about the NLP features. This may be due to the simplicity of the text used for the study. However, we found that most participants were using the NLP-based annotations without realizing it. In most of the diagrams,

the text within each node was more concise when compared to the probe study and more often corresponded to the 'hinted' recommendation. Two participants suggested an automated paraphrasing feature (e.g., "air is very cold" to "cold air"). We believe that more targeted evaluations in a classroom setting would further validate our features (and suggest additional extensions). However, the results of our study demonstrate the effectiveness of texSketch and AD for creating consolidated, diagrammatic representations that support readers.

SCALING TEXSKETCH FOR OTHER DIAGRAM TYPES
Educational science texts are often causal in nature [62], and are also hard to understand [59]. Hence, thus far, we have focused our use case for AD on creating causal diagrams while working with science texts. However, we designed texSketch's architecture to support other types of structured diagrams as well as *unstructured* visual note-taking (i.e., sketch-noting [53]). We briefly describe how we extended texSketch to concept-maps and general visual note-taking.

Concept Maps: Concept maps are similar in structure to causal diagrams (i.e., nodes connected by edges) with the exception that edges are not directional, and concepts can be organized in a hierarchy. In our implementation, readers can select topic level concepts by using an $\widehat{ellipse}$ annotation, and sub-concepts using <u>underline</u> annotation. Concepts can be linked by drawing a connecting line between them. We use mermaid.js [48] to generate concept maps, and SpaCy's [2] entity extraction features to support NLP-based concept selection. However, this can be replaced with more advanced topic modeling libraries in the future.

Visual Note-taking: texSketch supports unconstrained note taking by combining text with visual elements (see Figure 6). Readers can direct ==copy portions of the text== into the diagram view by using the highlighter tool. In addition we offer three glyph (diagram element) alternatives that were common in examples of visual notes we collected: (1) plain text labels/annotations, (2) iconic representation from a clip-art catalog, and (3) user-drawn glyphs. Readers can cycle between representations by repeatedly underlining the text (Figure 6): When underlining the text for the first time, texSketch adds that text as-is onto the diagram view (e.g., $\boxed{\text{sun}}$). A second underline will switch the representation to an icon corresponding to the text if one is available (e.g., ✿). To search for icons based on annotated text, we integrated with the Noun Project [57]. A third underline will allow the reader to define or draw their own glyph (future annotations of that word will use the custom glyph). Readers can draw on the diagram view, and freely arrange the text and diagram elements to support their creative note taking process.

Other diagram types can be readily added using our extensible architecture. For example, texSketch can support timeline visualizations by ordering event 'nodes' in a linear structure by using NLP features that detects events (e.g., [40]). Further, in many domains, learners benefit from visuospatial thinking (e.g., [76]), and texSketch can facilitate learning by adding domain specific annotation conventions, diagram templates, and representations (e.g., genetic structure diagrams for biology,
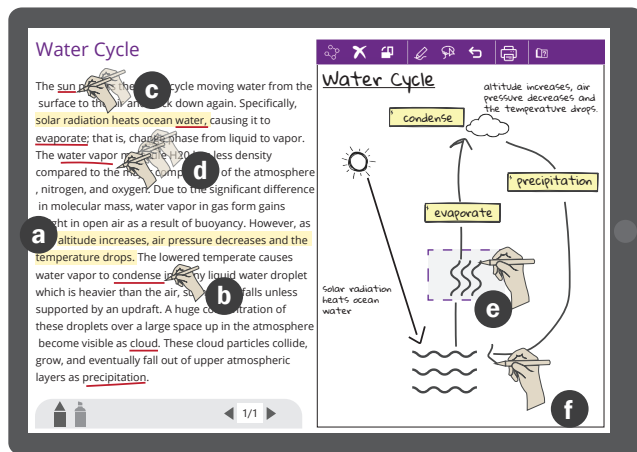
Figure 6. Visual note taking: (a) Highlighting text copies it to the diagram view. (b) Underlining once creates a node with that text. (c) A second underline changes it to an icon, and (d-e) a third underline lets readers define their own glyph for that text. (f) Readers can also freely draw on the diagram view.

or molecular structure in chemistry). When reading history, philosophy, and law, learners synthesize knowledge from multiple documents. In these cases, texSketch could support visual comparison of different arguments, and assist in counterfactual analysis. In the data visualization community, there is ongoing work to facilitate active reading for data visualization [74, 61]. The approaches have readers trace and annotate insights over different chart objects. In cases where visualizations are supplemented by text descriptions (e.g., news articles, scrollytelling visualizations, etc.), texSketch could support rich pen-and-ink interactions between text and chart views or even allow readers to generate chart elements while reading.

DISCUSSION
Text remains the primary medium for knowledge acquisition, and readers have to internalize complex structural relationships from linear text representations. With *Active Diagramming,* we have demonstrated an approach to comprehension that builds on well established active reading techniques by adding graphical representations to organize knowledge. This approach facilitates the externalization of mental models through diagramming. Based on our evaluation and participants' feedback, it is evident that the act of creation helps readers develop conceptual understanding that is critical to problem-solving, future forecasting, and learning transfer.

Our current prototype primarily supports underline type annotations and diagram structures. However, active reading involves a wider range of strategies. Different strategies may be appropriate based on preferred learning style or context (grade level, domain, type of material, pedagogy versus andragogy contexts, etc.). We believe our approach is scalable for different low and intermediate level representations. For example, instead of highlighting and annotating over raw text, readers could use pen and touch interaction to gather information into list structures (e.g., LiquidText [71]). Alternately, readers can use margin braces and glyph-type annotations to directly organize information over raw text, and then build di-

agrams based on those annotations. However, the right type of design requires additional study. Further, our current workflow requires users to set their learning intent by selecting a type of diagram prior to AD (i.e., goal directed learning [72]). To support multiple learning goals, future work can explore ways to easily switch between different diagram types. There is also significant opportunity to integrate other kinds of NLP approaches into texSketch. The specific types will depend on the kinds of diagrams created. Beyond support for individual readers, there are 'collaborative' active reading approaches where a community works to understand the material. We are interested in how texSketch can improve the learning experience for groups through social note-taking tools (e.g., nb [79]).

In our current evaluation, we emphasized effective diagram construction and did not focus on measuring learning outcomes. However, one of the most likely targets for texSketch is to support learning in classroom settings. We are currently working with instructors and middle-school students to deploy texSketch in the classroom to assess learning objectives and intervention strategies. Further, while we have struck a balance between different types of cognitive loads (the primary comprehension task and secondary construction tasks), other options may be more suitable for specific learning contexts. For example, an adult graduate student reading a paper may want more automation because they are building from significant prior knowledge and no longer benefit from constructing a diagram from scratch. An adult learner may make use of a much larger gestural language to build more complex diagrams. We have created texSketch with this possibility in mind, and additional research can help identify variants adapted for different kinds of learning.

CONCLUSION
Based on models for science text comprehension, we present an end-to-end workflow for knowledge externalization called Active Diagramming. Through a probe study, we identified key features needed to support active diagramming. The result is texSketch, a pen-and-ink system for building diagrams during reading. With texSketch, we focus on helping the reader convert annotations they make in text into integrated mental models. The iterative process of select-organize-integrate lets readers take text-level annotations into connected propositions and then combine them. NLP-based features on key terms help focus the reader's attention. Our interactions implement a consistent gesture language and allow readers flexibility in switching between text and diagram representations as they read. Our approach demonstrates a balance between providing readers with opportunities to learn through active diagramming while reducing non-desirable and distracting difficulties.

ACKNOWLEDGMENTS

REFERENCES

[1] Mortimer J Adler and Charles Van Doren. 2014. *How to read a book: The classic guide to intelligent reading*. Simon and Schuster.

[2] Explosion AI. 2019. spaCy. (2019). `https://spacy.io/`

[3] Shaaron Ainsworth. 2006. DeFT: A conceptual framework for considering learning with multiple representations. *Learning and instruction* 16, 3 (2006), 183–198.

[4] Lisa Anthony and Jacob O Wobbrock. 2010. A lightweight multistroke recognizer for user interface prototypes. In *Proceedings of Graphics Interface 2010*. Canadian Information Processing Society, 245–252.

[5] David Bargeron and Tomer Moscovich. 2003. Reflowing digital ink annotations. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 385–393.

[6] Aaron Bauer and Kenneth R Koedinger. 2007. Selection-based note-taking applications. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 981–990.

[7] Mohammed Belatar and François Coldefy. 2010. Sketched menus and iconic gestures, techniques designed in the context of shareable interfaces. In *ACM International Conference on Interactive Tabletops and Surfaces*. ACM, 143–146.

[8] Jared N Bott and Joseph J LaViola Jr. 2010. A pen-based tool for visualizing vector mathematics. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*. Eurographics Association, 103–110.

[9] Marius Brade, Christian Brändel, Angelika Salmen, and Rainer Groh. 2012. SketchViz: a sketching interface for domain comprehension tasks illustrated by an industrial network use case. In *Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies*. ACM, 30.

[10] Marius Brade, Florian Schneider, Angelika Salmen, and Rainer Groh. 2013. OntoSketch: Towards digital sketching as a tool for creating and extending ontologies for non-experts. In *Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies*. ACM, 9.

[11] Peter Brandl, Christoph Richter, and Michael Haller. 2010. Nicebook: supporting natural note taking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 599–608.

[12] Ann L Brown and Jeanne D Day. 1983. Macrorules for summarizing texts: The development of expertise. *Journal of verbal learning and verbal behavior* 22, 1 (1983), 1–14.

[13] Breanne J Byiers, Joe Reichle, and Frank J Symons. 2012. Single-subject experimental design for evidence-based practice. *American journal of speech-language pathology* (2012).

[14] Nicky Case. 2019. Loopy: a tool for thinking in systems. (2019). `https://ncase.me/loopy/`

[15] Nicholas Chen, Francois Guimbretiere, and Abigail Sellen. 2012. Designing a multi-slate reading environment to support active reading activities. *ACM Transactions on Computer-Human Interaction (TOCHI)* 19, 3 (2012), 18.

[16] Michelene TH Chi. 2009. Active-constructive-interactive: A conceptual framework for differentiating learning activities. *Topics in cognitive science* 1, 1 (2009), 73–105.

[17] Michelene TH Chi, Nicholas De Leeuw, Mei-Hung Chiu, and Christian LaVancher. 1994. Eliciting self-explanations improves understanding. *Cognitive science* 18, 3 (1994), 439–477.

[18] Travis J Cossairt and Joseph J LaViola Jr. 2012. SetPad: a sketch-based tool for exploring discrete math set problems. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*. Eurographics Association, 47–56.

[19] Nathalie Coté, Susan R Goldman, and Elizabeth U Saul. 1998. Students making sense of informational text: Relations between processing and representation. *Discourse Processes* 25, 1 (1998), 1–53.

[20] Richard Cox. 1999. Representation construction, externalised cognition and individual differences. *Learning and instruction* 9, 4 (1999), 343–363.

[21] Vonnie M DiCecco and Mary M Gleason. 2002. Using graphic organizers to attain relational knowledge from expository text. *Journal of learning disabilities* 35, 4 (2002), 306–320.

[22] Docear. 2019. The Academic Literature Suite. (2019). `http://www.docear.org/`

[23] Matthew W Easterday, Jordan S Kanarek, and Maralee Harrell. 2009. Design requirements of argument mapping software for teaching deliberation. *Online deliberation: Design, research, and practice* (2009), 317–23.

[24] Hugging Face. 2019. Neural Coreference. (2019). `https://huggingface.co/coref/`

[25] Kenneth Forbus, Jeffrey Usher, Andrew Lovett, Kate Lockwood, and Jon Wetzel. 2011. CogSketch: Sketch understanding for cognitive science research and for education. *Topics in Cognitive Science* 3, 4 (2011), 648–666.

[26] Janice D Gobert and John J Clement. 1999. Effects of student-generated diagrams versus student-generated summaries on conceptual understanding of causal and dynamic knowledge in plate tectonics. *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching* 36, 1 (1999), 39–53.

[27] Gene Golovchinsky, Morgan N Price, and Bill N Schilit. 1999. From reading to retrieval: freeform ink annotations as queries. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. Citeseer, 19–25.

[28] Ken Hinckley, Xiaojun Bi, Michel Pahud, and Bill Buxton. 2012. Informal information gathering techniques for active reading. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1893–1896.

[29] Ken Hinckley, Shengdong Zhao, Raman Sarin, Patrick Baudisch, Edward Cutrell, Michael Shilman, and Desney Tan. 2007. InkSeine: In Situ search for active note taking. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 251–260.

[30] Matthew Hong, Anne Marie Piper, Nadir Weibel, Simon Olberding, and James Hollan. 2012. Microanalysis of active reading behavior to inform design of interactive desktop workspaces. In *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*. ACM, 215–224.

[31] Walter Kintsch. 1988. The role of knowledge in discourse comprehension: A construction-integration model. *Psychological review* 95, 2 (1988), 163.

[32] Walter Kintsch and Teun A Van Dijk. 1978. Toward a model of text comprehension and production. *Psychological review* 85, 5 (1978), 363.

[33] Walter Kintsch and CBEMAFRS Walter Kintsch. 1998. *Comprehension: A paradigm for cognition*. Cambridge university press.

[34] David Kirsh. 2010. Thinking with external representations. *Ai & Society* 25, 4 (2010), 441–454.

[35] Bongshin Lee, Timothy Dwyer, and Nathalie Henry Riche. 2017. Authoring visual representations for text-based documents. (April 20 2017). US Patent App. 14/945,869.

[36] Youngmin Lee and David W Nelson. 2004. A conceptual framework for external representations of knowledge in teaching and learning environments. *Educational Technology* (2004), 28–36.

[37] Jose A León and Inmaculada Escudero. 2015. Understanding causality in science discourse for middle and high school students. Summary task as a strategy for improving comprehension. In *Improving reading comprehension of middle and high school students*. Springer, 75–98.

[38] Jimmie Leppink, Fred Paas, Cees PM Van der Vleuten, Tamara Van Gog, and Jeroen JG Van Merriënboer. 2013. Development of an instrument for measuring different types of cognitive load. *Behavior research methods* 45, 4 (2013), 1058–1072.

[39] Chunyuan Liao, François Guimbretière, Ken Hinckley, and Jim Hollan. 2008. Papiercraft: A gesture-based command system for interactive paper. *ACM Transactions on Computer-Human Interaction (TOCHI)* 14, 4 (2008), 18.

[40] Xiao Ling and Daniel S. Weld. 2010. Temporal Information Extraction. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI'10)*. AAAI Press, 1385–1390. `http://dl.acm.org/citation.cfm?id=2898607.2898828`

[41] Robert F Lorch. 1989. Text-signaling devices and their effects on reading and memory processes. *Educational psychology review* 1, 3 (1989), 209–234.

[42] Zhicong Lu, Mingming Fan, Yun Wang, Jian Zhao, Michelle Annett, and Daniel Wigdor. 2019. InkPlanner: Supporting Prewriting via Intelligent Visual Diagramming. *IEEE transactions on visualization and computer graphics* 25, 1 (2019), 277–287.

[43] Arnold M Lund. 2001. Measuring usability with the use questionnaire12. *Usability interface* 8, 2 (2001), 3–6.

[44] MarginNote. 2019. MarginNote Software. (2019). `https://www.marginnote.com`

[45] Catherine C Marshall. 1997. Annotation: from paper books to the digital library. In *Proceedings of the second ACM international conference on Digital libraries*. ACM, 131–140.

[46] Richard E Mayer. 1984. Aids to text comprehension. *Educational psychologist* 19, 1 (1984), 30–42.

[47] Hrim Mehta, Adam Bradley, Mark Hancock, and Christopher Collins. 2017. Metatation: Annotation as implicit interaction to bridge close and distant reading. *ACM Transactions on Computer-Human Interaction (TOCHI)* 24, 5 (2017), 35.

[48] MermaidJS. 2019. mermaid. (2019). `https://mermaidjs.github.io/`

[49] Bonnie JF Meyer and Leonard W Poon. 2001. Effects of structure strategy training and signaling on recall of text. *Journal of educational psychology* 93, 1 (2001), 141.

[50] Thomas P Moran, Patrick Chiu, and William Van Melle. 1997. Pen-based interaction techniques for organizing material on an electronic whiteboard. In *ACM Symposium on User Interface Software and Technology*. 45–54.

[51] Meredith Ringel Morris, AJ Bernheim Brush, and Brian R Meyers. 2007. Reading revisited: Evaluating the usability of digital display surfaces for active reading tasks. In *Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP'07)*. IEEE, 79–86.

[52] Judith S Olson and Wendy A Kellogg. 2014. *Ways of Knowing in HCI*. Vol. 2. Springer.

[53] Karin Perry, Holly Weimar, and Mary Ann Bell. 2017. *Sketchnoting in school: discover the benefits (and fun) of visual note taking*. Rowman & littlefield.

[54] Héctor R Ponce, Richard E Mayer, M Soledad Loyola, and Mario J López. 2019. Study Activities That Foster Generative Learning: Notetaking, Graphic Organizer, and Questioning. *Journal of Educational Computing Research* (2019), 0735633119865554.

[55] Morgan N Price, Gene Golovchinsky, and Bill N Schilit. 1998a. Linking by Inking: Trailblazing in a Paper-Like Hypertext.. In *HyperText*. 30–39.

[56] Morgan N Price, Bill N Schilit, and Gene Golovchinsky. 1998b. XLibris: The active reading machine. In *Conference on Human Factors in Computing Systems: CHI 98 conference summary on Human factors in computing systems*, Vol. 18. 22–23.

[57] Noun Project. 2019. Noun Project - Icons for Everything. (2019). `https://thenounproject.com/`

[58] Yann Riche, Nathalie Henry Riche, Ken Hinckley, Sheri Panabaker, Sarah Fuelling, and Sarah Williams. 2017. As We May Ink?: Learning from Everyday Analog Pen Use to Improve Digital Ink Experiences.. In *CHI*. 3241–3253.

[59] Donald J Richgels, Lea M McGee, Richard G Lomax, and Catherine Sheard. 1987. Awareness of four text structures: Effects on recall of expository text. *Reading Research Quarterly* (1987), 177–196.

[60] Daniel H Robinson and Kenneth A Kiewra. 1995. Visual argument: Graphic organizers are superior to outlines in improving learning from text. *Journal of educational psychology* 87, 3 (1995), 455.

[61] Hugo Romat, Nathalie Riche, Ken Hinckley, Bongshin Lee, Caroline Appert, Emmanuel Pietriga, and Christopher Collins. 2019. ActiveInk:(Th) Inking with Data. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM.

[62] Wesley C Salmon. 1998. *Causality and explanation*. Oxford University Press.

[63] Mike Scaife and Yvonne Rogers. 1996. External cognition: how do graphical representations work? *International journal of human-computer studies* 45, 2 (1996), 185–213.

[64] Bill N Schilit, Gene Golovchinsky, and Morgan N Price. 1998. Beyond paper: supporting active reading with free form digital ink annotations. In *CHI*, Vol. 98. Citeseer, 249–256.

[65] Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. 2014. Diagram understanding in geometry questions. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

[66] Preet Shihn. 2019. Rough.js Create graphics with a hand-drawn, sketchy, appearance. (2019). `https://roughjs.com/`

[67] Frank Shipman, Morgan Price, Catherine C Marshall, and Gene Golovchinsky. 2003. Identifying useful passages in documents based on annotation patterns. In *International Conference on Theory and Practice of Digital Libraries*. Springer, 101–112.

[68] Antonio Sorgente, Giuseppe Vettigli, and Francesco Mele. 2013. Automatic Extraction of Cause-Effect Relations in Natural Language Text. *DART@ AI* IA* 2013 (2013), 37–48.

[69] Craig J Sutherland, Andrew Luxton-Reilly, and Beryl Plimmer. 2016. Freeform digital ink annotations in electronic documents: A systematic mapping study. *Computers & Graphics* 55 (2016), 1–20.

[70] Ivan E Sutherland. 1964. Sketchpad a man-machine graphical communication system. *Simulation* 2, 5 (1964), R–3.

[71] Craig S Tashman and W Keith Edwards. 2011. LiquidText: a flexible, multitouch environment to support active reading. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 3285–3294.

[72] P Van Meter and CM Firetto. 2013. Cognitive model of drawing construction. *Learning through visual displays* (2013), 247–280.

[73] Peggy Van Meter and Joanna Garner. 2005. The promise and practice of learner-generated drawing: Literature review and synthesis. *Educational Psychology Review* 17, 4 (2005), 285–325.

[74] Jagoda Walny, Samuel Huron, Charles Perin, Tiffany Wun, Richard Pusch, and Sheelagh Carpendale. 2018. Active reading of visualizations. *IEEE transactions on visualization and computer graphics* 24, 1 (2018), 770–780.

[75] Jo Wood, Petra Isenberg, Tobias Isenberg, Jason Dykes, Nadia Boukhelifa, and Aidan Slingsby. 2012. Sketchy rendering for information visualization. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2749–2758.

[76] Hsin-Kai Wu and Priti Shah. 2004. Exploring visuospatial thinking in chemistry learning. *Science education* 88, 3 (2004), 465–492.

[77] Sally PW Wu and Martina A Rau. 2019. How Students Learn Content in Science, Technology, Engineering, and Mathematics (STEM) Through Drawing Activities. *Educational Psychology Review* (2019), 1–34.

[78] Dongwook Yoon, Nicholas Chen, and François Guimbretière. 2013. TextTearing: opening white space for digital ink annotation. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, 107–112.

[79] Sacha Zyto, David Karger, Mark Ackerman, and Sanjoy Mahajan. 2012. Successful classroom deployment of a social document annotation system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1883–1892.